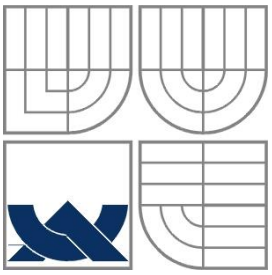
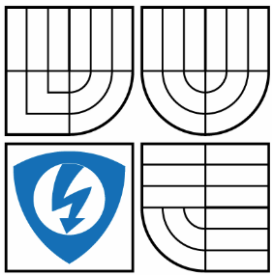


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

PROTOKOL CAN PRO ŘÍZENÍ

CAN PROTOCOL FOR CONTROL

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — TOMÁŠ PAVLIŠIN
AUTHOR

VEDOUCÍ PRÁCE — Ing. ONDŘEJ HYNČICA
SUPERVISOR

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Bakalářská práce

bakalářský studijní obor

Automatizační a měřicí technika

Student:	Tomáš Pavlišin	***ID:***	154829
Ročník:	3	***Akademický rok:***	2014/2015

NÁZEV TÉMATU:

Protokol CAN pro řízení

POKYNY PRO VZPRACOVÁNÍ:

Cílem práce je seznámení se s protokolem CAN, jeho využitím v automatizaci a s implementací komunikace do mikrokontroléru.

1) Seznamte se s protokolem CAN a s jeho rozšiřujícími a nástavbovými variantami, zejména CANopen.
2) Implementujte obsluhu CAN rozhraní do mikrokontroléru a vytvořte aplikaci pro testování komunikace.
3) Proveďte měření rychlosti komunikace v závislosti na vzdálenosti a komunikační rychlosti.
4) Vytvořte jednoduchou aplikaci, která bude realizovat komunikaci s průmyslovým CAN zařízením.

DOPORUČENÁ LITERATURA:

SLOSS, Andrew N, Dominic SYMES a Chris WRIGHT. ARM system developer´s guide: designing and optimizing system software. Amsterdam: Elsevier, 2004, 689 s. ISBN 15-586-0874-5.

Termín zadání:	9.2.2015	***Termín evzdání:***	***25.5.2015***

Vedoucí práce: Ing. Ondřej Hynčica

Konzultanti bakalářské práce:

doc. Ing. Václav Jirsík, CSc.

Předseda oborové rady

UPOZORNĚNÍ:

Abstrakt

Cieľom tejto práce je riadenie motorového pohonu s implikovaným CAN modulom. Riadenie realizuje vývojová doska od firmy NXP LPC11C24 pomocou komunikácie po zbernici CAN s rozširujúcou nadstavbou CANopen. Práca obsahuje aj software pre mikroprocesor riadiaci priemyslové zariadenie VTA-DOOR, aplikáciu na testovanie komunikácie a taktiež užívateľský software pre PC, ktorým môžeme meniť parametre riadenia.

Kľúčové slová

CAN, Controller Area Network, CAN v Automatizácii (CiA), CANopen, zbernica, FRDM-KE06Z, LPC11C24

Abstract

The goal of this thesis is to control motor drive with implemented CAN module. The control is carried out by development board from NXP company LPC11C24, through CAN bus communication with expansion extension CANopen. The thesis also contains microprocessor software controlling industrial device VTA-DOOR, application to test communication and user program for PC which is used to changing control parameters.

Keywords

CAN, Controller Area Network, CAN in Automation (CiA), CANopen, fieldbus, FRDM-KE06Z, LPC11C24

Bibliografická citácia:

PAVLIŠIN, T. *Protokol CAN pro řízení*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2013. 45s. Vedúci bakalárskej práce bol Ing. Ondřej Hynčica

Prehlásenie

„Prehlasujem, že svoju bakalársku prácu na tému Protokol CAN pro řízení som vypracoval samostatne pod vedením vedúceho bakalárskej práce a s použitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedené bakalárskej práce ďalej prehlasujem, že v súvislosti s vytvorením tejto bakalárskej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona č. 121/2000 Sb., vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovení časti druhej, hlavy VI. diel 4 Trestného zákonníku č. 40/2009 Sb.

V Brne dňa: **25. mája 2015** ……

podpis autora

Poďakovanie

Ďakujem spoločnosti Beta Control s.r.o. a predovšetkým vedúcemu bakalárskej práce Ing. Ondřejovi Hynčicovi za účinnú metodickú, pedagogickú a odbornú pomoc a ďalšie cenné rady pri spracovaní mojej bakalárskej práce.

V Brne dňa: **25. mája 2015** ……

podpis autora

Obsah

1. Úvod

Bakalárska práca sa v prvom rade zaoberá oboznámením sa s protokolom CAN a CANopen a ich využití v riadení. Cieľom práce je vytvorenie riadenia priemyselného zariadenia s implikovaným CAN budičom. Toto zariadenie bude riadené pomocou mikrokontroléra cez komunikáciu CAN s nadstavbou CANopen. Táto práca obsahuje históriu a popis protokolu CAN a CANopen, možnosti zakončenia zbernice, komunikáciu po zbernici, riešenie kolízií, popis vývojových prostredí, testovanie komunikácie a výslednú ukážkovú aplikáciu.

Oboznámením sa s fyzickou a linkovou vrstvou protokolu CAN a rešerší vývojových mikrokontrolérov s CAN budičom plán práce začína. Pokračuje inštaláciou a nastavením softvérov na programovanie mikroprocesorov a nastavenie komunikačného rozhrania OpenSDA mikrokontroléra kvôli možnosti debuggingu. Nastavením cpu, rozhrania CAN a sériovej komunikácie UART sa dostaneme k možnosti vytvoriť komunikáciu po zbernici CAN medzi dvoma a viacerými zariadeniami a komunikáciu s PC. Nasleduje oboznámenie sa s aplikačnou vrstvou čo je rozširujúca nadstavba - CANopen. Pomocou objektov CANopen realizovať komunikáciu s priemyslovým zariadením VTA-DOOR. Následne otestovať funkčnosť a parametre komunikácie a vyhodnotiť ich.

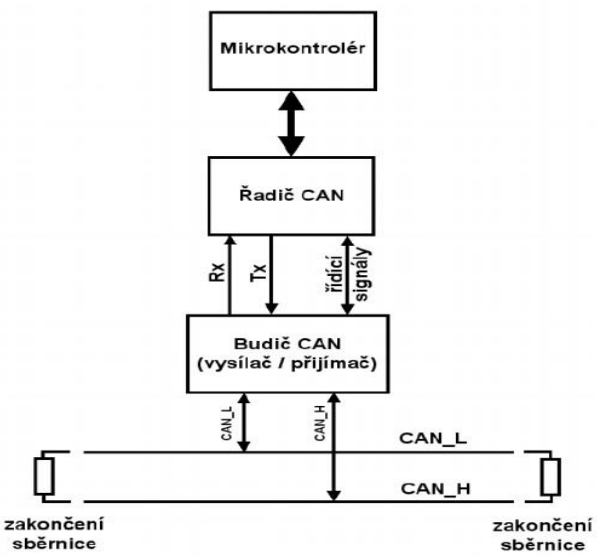
2. Popis protokolu CAN a CANopen

CAN alebo Controller Area Network, môžeme preložiť ako dátová zbernica miestnej siete. Bol vytvorený spoločnosťou BOSCH v roku 1983 pre použitie v automobilovom priemysle, ktorý koncom osemdesiatych rokov prešiel na digitálne riadenie. Z toho dôvodu bolo nutné vyvinúť protokol pre diagnostiku týchto systémov. Pôvodným zámerom bola predovšetkým úspora kabeláže a zabezpečenie prenosu informácií medzi snímačmi, riadiacimi a výkonovými prvkami v automobiloch. Ako protokol bol oficiálne vyhlásený roku 1986. Nasledujúci rok sa na trhu objavili prvé CAN budiče predstavené firmou Philips Semiconductors. V roku 1991 docielila podobu základnej verzie CAN 2.0, ktorá sa modifikovala vývojom na systémy CAN 2.0A a CAN 2.0B, navzájom kompatibilných. Aplikácia protokolu sa vyskytla v prvých modeloch automobilov Mercedes-Benz roku 1992. Do medzinárodného štandardu ISO 11898 sa protokol CAN 2.0A preniesol roku 1993. Norma definuje dátovú a fyzickú vrstvu zbernice do prenosovej rýchlosti 1Mbit/s. O dva roky neskôr bola vytvorená nová špecifikácia CAN 2.0B, ktorá definovala dva formáty správy, standard a extended(veľkosť identifikátoru správy sa líši, standard – 11bit, extended – 29bit). Pre svoju jednoduchosť, spoľahlivosť, odolnosť pri extrémnych podmienkach (rušenie, teplota, atd.), relatívne vysokú prenosovú rýchlosť, dobré zabezpečenie proti chybám a nízku cenu sa stala zbernica CAN veľmi populárna medzi fieldbusy. Preto sa tento protokol čoraz častejšie začal využívať aj v oblastiach riadiacej techniky. [1][2]

Aj keď bol CAN pôvodne vyvinutý na používanie v osobných automobiloch, prvá aplikácia prišla z iných častí trhu. Vo svojich začiatkoch bol CAN veľmi populárny hlavne v severnej Európe. Začiatkom roku 1992, niekoľko spoločností založilo neziskovú organizáciu CiA (CAN in Automation), s cieľom poskytnúť technické, produktové a marketingové informácie. Týmto krokom chceli zviditeľniť CAN a poskytnúť cestu pre budúci vývoj tohto protokolu. Jedným z prvých úloh CiA bolo špecifikovanie aplikačnej vrstvy CAN (CAN Application Layer - CAL). Hoci prístup CAL bol akademicky správny a bolo ho možné použiť v priemyslových aplikáciách, každý užívateľ potreboval navrhnúť nový profil. Od roku 1993 a v rámci projektu EEC ESPRIT III - ASPIC, európske konzorcium vedené firmou Bosch vyvinulo prototyp budúceho CANopen, založeného na CAL profile pre vstavané siete v produkčných bunkách. V roku 1995 CiA vydala úplne prepracovaný CANopen komunikačný profil. Profil CANopen definuje rámec pre programovateľné systémy ako aj rôzne zariadenia, rozhrania a aplikácie. To bol dôležitý dôvod, prečo celé priemyselné segmenty (napríklad. námorné aplikácie, lekárske systémy, tlačiarenské stroje a iné) sa rozhodli použiť CANopen. Členovia CiA postupne definujú CANopen aplikačné profily, ktoré špecifikujú všetky rozhrania zariadení používané v konkrétnej aplikácii. V súčasnosti je tento protokol súčasťou normy EN 50325-4. Cieľové využitie je v aplikáciách priemyslovej automatizácie, kde sú zariadenia ako napr. pohony, serva, I/O moduly apod., ktoré komunikujú cez zbernicu CAN. Jedny z hlavných vlastností protokolu CANopen je otvorenosť a zároveň dôsledne zachovanie kompatibility.

2.1. Základné vlastnosti CAN protokolu

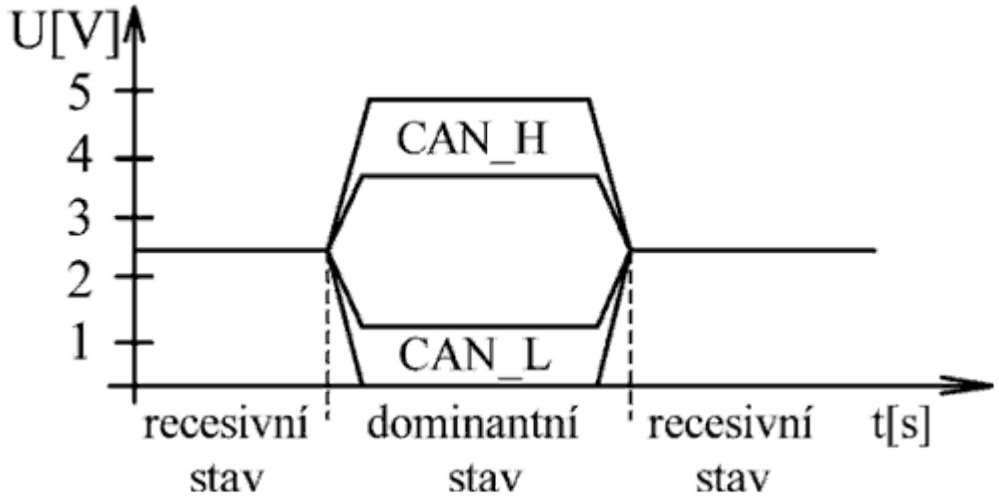
CAN protokol je multiplexná sériová komunikácia vysokej prenosovej rýchlosti so zaistením vernosti dát. Jednotlivé komunikačné uzly (zariadenia) musia byť medzi sebou fyzicky spojené. Po zbernici prebieha komunikácia medzi dvoma uzlami pomocou správ (dátová správa, žiadosť o data). Môže pracovať v dvoch režimoch. Každý uzol môže byť master a tým pádom riadiť zvyšné uzly zbernice (multi-master). Z toho vyplýva zjednodušenie riadenia a zvýšenie spoľahlivosti oproti riadeniu z jedného nadradeného uzla (master-slave). Zariadenie, ktoré chce vysielať, vyšle správu do siete a stane sa master po dobu odoslania správy. Po odoslaní správy uvoľní sieť pre ďalšie zariadenia. Uzly, ktoré "počúvajú" na zbernici, poznajú, či je správa určená pre ne alebo ju majú ignorovať podľa identifikátora (ID), ktorý správa obsahuje. Správy, ktoré má daný uzol prijať zaobstaráva filter. Identifikátor taktiež určuje prioritu daného zariadenia. Priorita sa využíva v okamžiku, kedy začnú vysielať dve zariadenia v tú istú dobu, prednosť má to s nižším ID (riešenie kolízií). Pomocou špeciálnych správ (chybové správy a správy o preťažení) je zaistený management siete. CANom je možné vybudovať sieť s rozličným prúdom, napätím, svetlom atd. Dôležité je venovať pozornosť impedančnému prispôsobeniu a ochrane na vyžiarené i vžiarené signály. Množstvo zariadení pre komunikáciu je teoreticky neobmedzené, reálne je limitované výkonovým zaťažením siete a možnosťami jednotlivých zariadení. Veľká výhoda je možnosť pozmenenia vytvorenej siete uberaním alebo pridávaním zariadení v sieti. Napríklad pri potrebe odstavenia zdroja chybných správ, alebo pridanie uzlu pre diagnostiku systému.



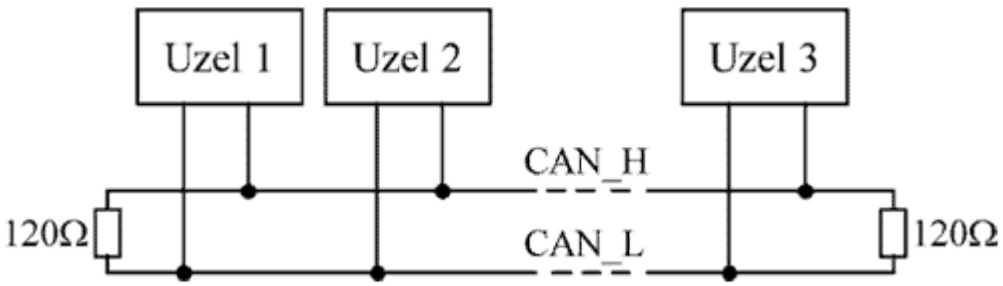
Obrázok 1: Príklad schematickej konfigurácie jednoduchej siete [9][10]

3. Fyzická vrstva

Protokol CAN má na zbernici definované dva stavy: dominant (dominantný predstavuje stav log.0) a recessive (recesivný, log.1). Základná požiadavka na túto vrstvu je realizácia funkcie logického súčinu. To znamená, že pokiaľ aspoň jedno zariadenie bude v stave dominant (aktívny), tak výsledný stav na zbernici bude dominant. Stav recessive (pasívny) nastane len v okamihu, keď všetky zariadenia vysielajú stav recessive. Diferenciálna zbernica definovaná podľa normy ISO 11898 sa najčastejšie využíva pre realizáciu fyzického prenosového média. Zbernicu tvoria dva signálové vodiče (CAN_H a CAN_L). Rozdielovým napätím týchto dvoch vodičov je definovaná úroveň na zbernici. Úroveň dominant má nenulovú hodnotu rozdielového napätia, úroveň recessive hodnotu U_{roz} = 0V. Konštrukciou snímačov signálových vodičov je v dominantnom stave zaistené napätie na vodiči CAN_H v rozmedzí 3,5 až 5V, na vodiči CAN_L 0 až 1,5V. V stave recessive je zaistené odporovou sieťou na vstupe prijímača napätia na oboch vodičoch rovnaké. [3]



Obrázok 2: Tolerančné pásmo napäťových úrovní logických stavov na zbernici CAN[3]

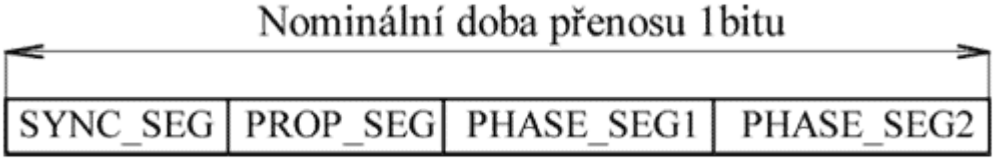


Obrázok 3: Principiálna schéma zbernice CAN[3]

Zakončovacími odpormi rovnakej veľkosti (najčastejšie používaná hodnota 120Ω) na oboch koncoch zbernice eliminujeme odrazy vedenia.

3.1. Synchronizácia a časovanie

Pre udržanie synchronizácie počas prenosu správ medzi uzlami CAN sa používajú zmeny úrovne signálu na zbernici. Doba trvanie jedného informačného bytu sa delí na štyri časové segmenty. Synchronizačný, propagačný a dva fázové segmenty. Každý segment sa delí na časové kvanta. Časové kvantum t_q je najmenšia jednotka, odvodená z frekvencie kryštálu. Dĺžka jedného bitu je rovná súčtu všetkých časových kvant (doba prenosu). Doba prenosu môže
Segmenty:
SYNC_SEG – synchronizácia rôznych modulov na zbernici. Očakáva sa hrana začiatku odosielaného signálu. Veľkosť tohto segmentu je rovný 1 t_q.
PROP_SEG – kompenzácia meškania signálu spôsobeného dĺžkou vedenia zbernice.
$t_q = 1 – 8$.
PHASE_SEG1, PHASE_SEG2 – kompenzácia fázových chyb na zbernici. Medzi týmito segmentmi sa nachádza vzorkovací bod stavu zbernice. t_q = 1- 8 pre každý segment.

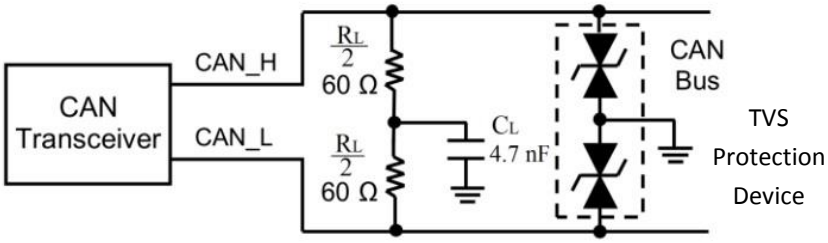


Obrázok 4: Prenos jedného informačného bitu v sieti CAN[3]

Dĺžka časových kvant sa mení pokiaľ očakávaná hrana odosielaného signálu sa vyskytne mimo synchronizačný segment. Túto dĺžku je možné meniť o ich programovateľný počet. Aby spôsob tejto kompenzácie bol realizovateľný bez vplyvu na obsah prenášaných správ, je využitá metóda doplnenia bitu s opačnou polaritu. Ak správa obsahuje 5bitov s rovnakou polaritou, automaticky sa zaradí do reťazca bitov bit s opačnou polaritou. Na prijímacej strane sa tento bit opäť vyradí. [3]

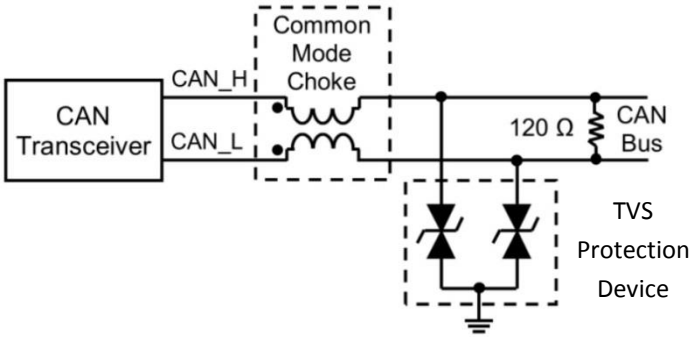
3.2. Zakončenie zbernice a dĺžka vedenia

Ako už bolo vyššie naznačené využitie zakončovacích rezistorov sa využíva na elimináciu odrazov na zbernici. Rezistory sa volia v rozmedzí 95 Ω až 140 Ω. Norma odporúča použitie tienených vodičov, kvôli lepšej EMC imunite. Pre normálne použitie to nie je nevyhnutné. Tiež je možné zakončenie split termináciou. Tá je vhodná pre stabilizáciu napätia na zbernici v stave recessive, a tiež ako dolnú pásmovú priepust. Pre zvýšenie elektromagnetickej imunity a kompatibility je možné použiť toto zapojenie, pri použití rovnakých rezistorov.



Obrázok 5: Split terminácia[5]

Pre vytvorenie robustného systému, je vhodné skombinovať viacero prvkov. Je možné použiť TVS diódu (Transil). Transil slúži k pohlteniu energie pri veľkej napäťovej špičke. Pre niektoré prípady je odporúčané použiť tlmivku rovnakého napätia (Common Mode Choke). Tieto diódy majú rýchlu dobu nábehu (pod 1ns), takže napätie na zbernici by nemalo prekročiť menovité napätie na CAN budiči. TVS diódy majú väčší PN prechod než zenerové diódy a namiesto stabilizácie napätia sú navrhnuté na prekonanie napäťových špičiek. TVS dióda je zobrazená na obrázku 5 a 6.



Obrázok 6: Použitie cievky na zbernici[5]

Použite Common Mode Choke zoslabuje superponovaný šum na oboch stranách dátovej linky. Cievky sú efektívne nástroje pre realizáciu filtrácie bez veľkého skreslenia. [5]

Dĺžka vedenia nám obmedzuje rýchlosť prenosu dát. Maximálne 40 metrovým vedením môžeme prenášať data maximálnou rýchlosťou 1 Mbit/s. Tieto hodnoty udáva norma. Pravidlo pre maximálnu povolenú rýchlosť komunikácie je:

Rýchlosť prenosu (Mbit/s) x Dĺžka zbernice ≤ 50

Tabuľka 1: Odporúčané komunikačné rýchlosti pri určitých dĺžkach vedenia

Dĺžka vedenia [m]	Maximálna rýchlosť prenosu [Mbit/s]
40	1,00
100	0,50
200	0,25
500	0,10
1000	0,05

4. Linková vrstva

Linková vrstva v protokole CAN je rozdelená ako v modeli ISO/OSI na podvrstvu **LLC** a **MAC**.

LLC (Logical Link Control) táto podvrstva riadi dátové spoje, čo znamená hlásenie o preťaženiach (Overload Notification) a filtráciu prijatých správ (Acceptance Filtering). Je zodpovedná za prenos bitov zo zdroja do cieľa.

MAC (Medium Access Control) reprezentuje jadro CAN protokolu. Vykonáva kódovanie dát, riadenie prístupu všetkých uzlov k médiu s vybranými prioritami správ, vkladanie doplnkových bitov do komunikácie (Stuffing/Destuffing), detekciu chýb a ich hlásenie a potvrdzovanie správne prijatých správ. [4]

4.1. Riadenie prístup k médiu a riešenie kolízií

Pokiaľ je zbernica v režime Bus free (voľná), môže akýkoľvek uzol začať vysielať. Prvý vysiela. Ostatné môžu vysielať až po odoslaní dát vysielajúceho uzla a ukončení komunikácie. Výnimku majú len chybové rámce, ktoré môžu vysielať hneď po zistení chyby ktoréhokoľvek uzla.

Pri súčasnom zahájení vysielania viacerými zariadeniami získa prístup na zbernicu ten, ktorý má najvyššiu prioritu (najnižšie ID). Identifikátor je uvedený na začiatku správy. Každý vysielač porovnáva hodnotu na zbernici s hodnotou práve vysielaného bitu. Ak zistí, že hodnota na zbernici je iná než tá, ktorú vysiela, okamžite preruší ďalšie vysielanie. Tým je zaistené, že nedôjde k poškodení správy s vyššou prioritou, a že bude odoslaná prednostne. Zariadenie, ktoré pri kolízii nedostalo prístup na zbernicu, musí vyčkať, kým zbernica nebude zase voľná (Bus free), a potom znovu vyslať správu. [2]

4.2. Zabezpečenie prenášaných dát

Prenášané správy protokolom CAN sú zabezpečené niekoľkými, súčasne prebiehajúcimi mechanizmami.

- monitoring
- CRC kód
- doplnenie bitu
- kontrola správy
- potvrdenie prijatej správy
-

Monitoring: vysielač porovnáva vyslanú hodnotu bitu s úrovňou na zbernici. Ak sú hodnoty rovnaké, vysielač pokračuje vo vysielaní. Pri detekcii rôznych hodnôt, pri práve prebiehajúcom riadení prístupu na zbernicu, sa preruší vysielanie. Prístup k médiu získa uzol s nižším ID. Pri detekcii rôznych hodnôt mimo riadenia prístupu na zbernicu a potvrdení prijatia správy, je vygenerovaná chyba bitu.

CRC kód: (Cyclic Redundancy Check) tvorí posledné pole odosielanej správy o veľkosti 15 bitov. CRC chyba je vygenerovaná, ak je táto chyba detekovaná ľubovoľným uzlom na zbernici. Chyba sa môže generovať vo všetkých bitoch správy podľa polynomu: $x^{15} + x^{14} + x^{10} + x^{8} + x^{7} + x^{4} + x^{3} + 1$, pretože sa nachádza na konci.

Doplnenie bytu: (bit stuffing) ak sa vysiela na zbernici za sebou idúcich päť bitov jednej úrovne, pridá sa do správy jeden bit opačnej úrovne. Slúži k detekcii chýb a k správnej časovej synchronizácii prijímačov jednotlivých zariadení. Chyba vkladania bitu sa vygeneruje pri jej detekcii.

Kontrola správy: (message frame check) chyba formátu správy. Pokiaľ sa na pozícii nejakého bitu objaví nepovolená hodnota, vygeneruje sa chyba rámca. Správa sa kontroluje podľa formátu uvedeného v špecifikácii.

Potvrdenie prijatia správy: (acknowledge) správne prijatú správu musí každé zariadenie pripojené na zbernicu potvrdiť. To platí aj pre zariadenia, ktoré danú správu neprijímajú. Zmenou bitu v poli ACK (bit 1) sa vykonáva potvrdenie. [1][2][4]

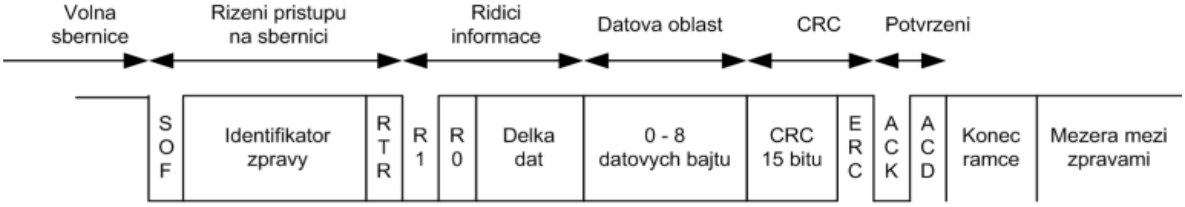
4.3. Prenos správ

V protokole CAN rozlišujeme podľa veľkosti dva druhy správ. **Standard** je formát správy s 11bitovým identifikátorom, definovaný podľa protokolu CAN 2.0A. **Extended** (rozšírený) definuje formát správy s 29bitovým identifikátorom definujúci špecifikáciu CAN2.0B. [3]

Základné bloky:
Data Frame: prenáša dátové informácie
Remote Frame: predáva žiadosť na prenos dát
Error Frame: odosiela sa pri detekovaní chyby na zbernici
Overload Frame: generuje zvláštne meškanie vkladané do predchádzajúcich blokov

4.3.1. Data Frame

Odosielanie dátovej správy je možné, len pri voľnej zbernici.



Obrázok 7: Dátová správa podľa špecifikácie CAN 2.0A[4]

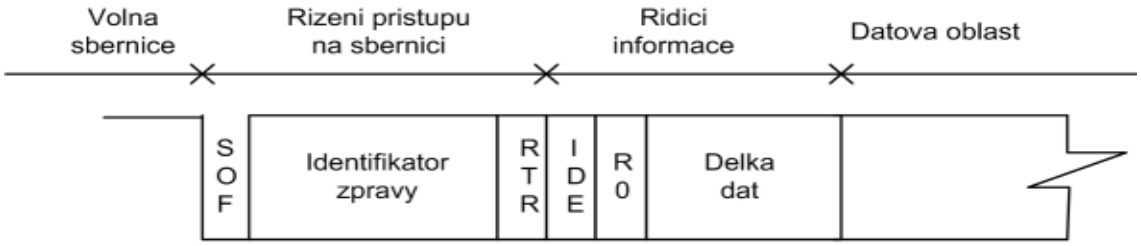
- **Začiatok správy**
  - SOF (1b) – Start of Frame - dominant
- **Riadenie prístupu k zbernici** (Arbitration Field)
  - Identifikátor správy (11b), určuje prioritu správy a jej význam
  - **RTR** (1b) – Remote Request, rozlíšenie správy (dátová – dominant (log 1), žiadosť o prístup k zbernici – recessive (log 0))
- **Riadiace informácie** (Control Field)
  - **R0**, **R1** (2b celkom) – rezervované bity
  - **Dĺžka dátovej správy** (4b) – počet prenášaných dátových bajtov správy. (hodnoty 0-8)
- **Dátová oblasť** (Data Field) – dátové bajty správy, max. 8Byte
- **CRC** (CRC Field) (16b), zabezpečovací CRC kód
  - **CRC** (15b)

- **ERC** (1b) – CRC oddeľovač,- dominant

- **Potvrdenie** (ACK Field) (2b)
  - **ACK**(1b) - potvrdenie
  - **ACD** (1b) – oddeľovač, recessive
- **Koniec správy**
  - **EOF** (7b) – End of Frame,– recessive
- **Medzera medzi správami** (3b) - (Interfame Space) – recessive

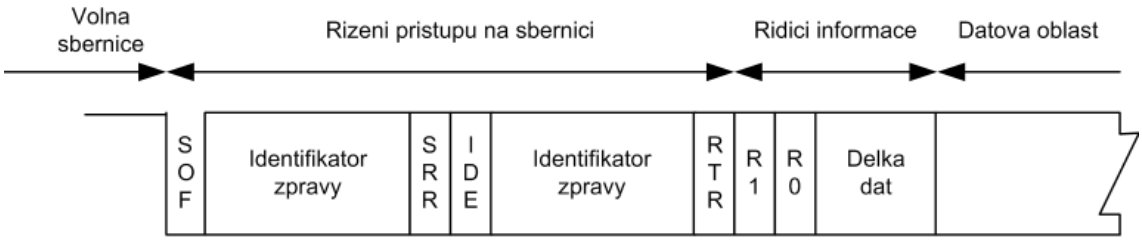
Formáty dátovej správy špecifikácie CAN 2.0B

Standard – prevzatý zo špecifikácie CAN 2.0A. Dĺžka identifikátoru je 11b. Jediný rozdiel je vo využití bitu R1. Ten indikuje, či je rámec štandardný alebo rozšírený. V CAN 2.0B sa tento bit označuje IDE (Identifier Extended). Ak je IDE v dominantom stave, jedná sa o formát standard a recessive pre formát extended.



Obrázok 8: Začiatok dátovej správy podľa špecifikácie CAN 2.0B (Standard)[4]

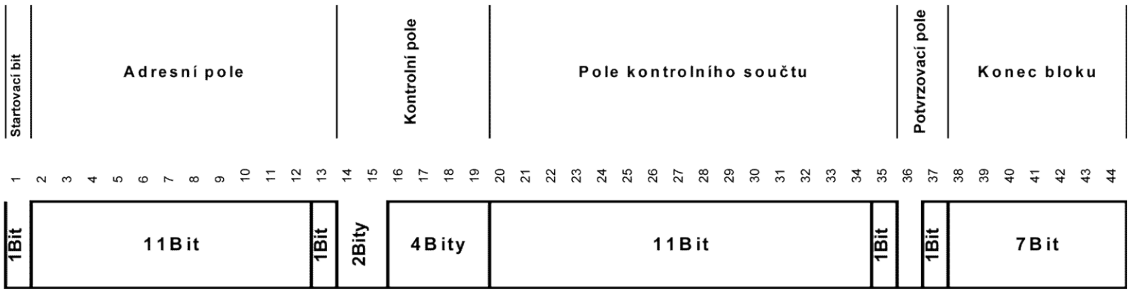
Extended – celková veľkosť identifikátora správy je 29b. Ten je rozdelený na dva časti, 11b (rovnaký identifikátor ako v štandardnom formáte) a 18b. RTR (Remote Request) bit je nahradený bitom SRR (Substitute Remote Request), ktorý má v tomto formáte vždy hodnotu recessive. To zaisťuje, aby získal prednosť štandardný rámec pri vzájomnej kolízii s rozšíreným formátom správy na jednej zbernici s rovnakým 11 bitovým identifikátorom. Bit IDE má vždy hodnotu recessive. RTR bit, ktorý udáva, či je to dátová správa alebo žiadosť o data, je presunutý na koniec druhej časti identifikátoru. Pre riadenie prístupu k médiu sú použité tieto zložky ID (11b), SRS, IDE, ID (18b), RTR. Priorita dátovej správy je určená v tomto poradí. [1][2][4]



Obrázok 9: Začiatok dátovej správy podľa špecifikácie CAN 2.0B (Extended)[4]

4.3.2. Remote Frame

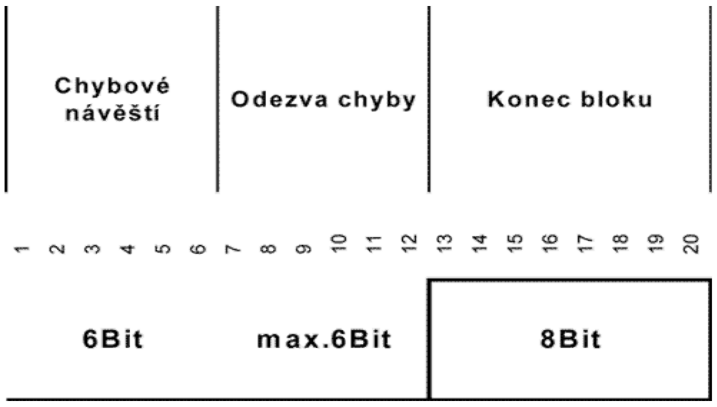
Tento formát rámca žiadosti o data je podobný ako formát data frame. Rozdiel je v chýbajúcej dátovej oblasti a bit RTR je nastavený do úrovne recessive. Princíp bloku je predať požiadavku príslušnému uzlu na predanie potrebných dát. [1][2]



Obrázok 10: Blok žiadosti o data[1]

4.3.3. Error Frame

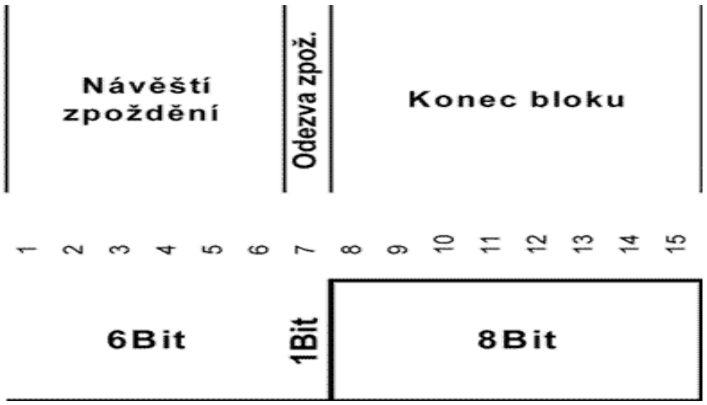
Slúži k signalizácii chýb na zbernici CAN. V momente, keď ľubovoľný uzol na zbernici detekuje v prenášanej správe chybu (bitu, CRC, rámca alebo chybu vloženia bitu), ihneď vygeneruje na zbernici chybový rámec (Error Frame). Chybový blok je zložený z troch častí. V prvej časti vyšle uzol analyzujúci chybu šesť po sebe idúcich logických núl (poruší sa bitová ochrana aktívnych uzlov – šiesty bit nie je inverzný k piatemu), ktoré začnú vysielať chybové bloky. Odozva v druhom bloku má maximálne 6b. Ukončovacia časť má 8b s tým, že zariadenia čakajú na prechod do úrovne 1 a následne pridá 7b. [1][2]



Obrázok 11: Chybový blok[1]

4.3.4. Overload Frame

Používa sa k predĺžení odozvy po ukončení bloku dát, chyby alebo výzvy. Má podobnú štruktúru ako chybový blok. Uzol vysiela tento ráme hlavne vtedy, keď potrebuje určitý čas na spracovanie predchádzajúcej správy. Obsahuje minimálne šesť dominantných bitov a následne sedem recesívnych. Vysielanie môže začínať až na konci správy, konci oddeľovača chýb alebo konci predchádzajúcej správy preťaženia. [1][2]



Obrázok 12: Správa o preťažení[1]

Každý uzol má zabudované dve interné počítadlá chýb. Tieto počítadlá udávajú pri príjmu a pri vysielaní. Hlásenie chýb uzla a jeho aktivita na zbernici môže podľa obsahu počítadiel meniť medzi troma stavmi (aktívny, pasívny, odpojený). Ak uzol generuje príliš veľa chýb, je automaticky odpojený.
Rozdelenie chýb z hľadiska hlásenia chýb:

- **Aktívne** (Error Active)
- **Pasívne** (Error Passive)
- **Odpojené** (Bus-off)

Aktívne – tieto uzly sa môžu aktívne podieľať na komunikácii po zbernici a v prípade, že detekujú ľubovoľnú chybu v práve prenášanej správe, vysielajú na zbernicu aktívny príznak chyby (Active Error Flag). Tento príznak je tvorený šiestimi po sebe idúcimi bitmi typu dominant, čím dôjde k poškodení prenášanej správy (porušenie pravidla vkladania bitov).

Pasívne – tieto uzly sa tiež podieľajú na komunikácii po zbernici, ale z hľadiska hlásenia chýb vysielajú len pasívny príznak chyby (Passive Error Flag). Ten je tvorený šiestimi bitmi typu recessive (nedôjde k zničeniu práve vysielanej správy).

Odpojené – tieto uzly nemajú žiaden vplyv na zbernicu, ich výstupné budiče sú vypnuté. [4]

5. Software a Hardware

5.1. Kinetis Design Studio (KDS)

Je to bezplatné integrované vývojové prostredie pre mikrokontroléry Kinetis, ktoré umožňuje robustné úpravy, zostavovanie a ladenie vlastných návrhov. Založený na voľnom, open-source software zahrnujúci Eclipse, GNU Compiler Collection (GCC), GNU Debugger (GDB), a iné. Návrhárom ponúka jednoduchý vývojový nástroj bez obmedzenia veľkosti kódu. Okrem toho, software Processor Expert, ktorý pomáha vytvárať výkonné aplikácie pomocou jeho vedomostnej základne len niekoľkými kliknutiami myšou. [11]

5.2. LPCXpresso

Je to značne integrované softvérové vývojové prostredie pre LPC ARM mikrokontroléry od firmy NXP, ktoré zahŕňa všetky nevyhnutné nástroje na rozvoj vysoko kvalitných softvérových aplikácií časovo a finančne efektívnym spôsobom. LPCXpresso je založený na Eclipse s veľkým množstvom špecifických vylepšení pre LPC. Je taktiež vybavený najnovšou verziou nástrojov GNU priemyslového štandardu s optimalizovanou knižnicou C, čím poskytuje profesionálne kvalitné nástroje za nízke ceny. LPCXpresso umožňuje návrhárom vyvíjať aplikácie od počiatočného hodnotenia až do finálneho produktu (end-to-end).

5.3. Mikroprocesor

Riadi komunikáciu modulu CAN a komunikáciu s počítačom cez zbernicu UART. Pre programovanie tohto procesora využívame developerské prostredie Kinetis Design Studio respektíve LPCXpresso.
Požiadavky na mikroprocesor:

- podpora protokolu CAN
- softwarová podpora
- dostupnosť na trhu
- jednoduchá obsluha
- cenová dostupnosť

V mojom prípade som vyberal medzi dvoma mikroprocesormi s CAN radičom. STM32F103 od spoločnosti ST (NUCLEO-F103RB) a MKE06Z128VLK spoločnosti Feescale Semiconductor (FRDM-KE06Z). Kvôli nízkej softwarovej podpore pre CAN protokol STMka som zvolil mikroprocesor MKE06Z128VLK. Tento mikroprocesor beží na jadre ARM Cortex M0. [7][8][9]

- veľkosť programovacej pamäte – 128 kB
- typ programovacej pamäte - Flash
- veľkosť dátovej RAM – 16 kB
- šírka dátovej zbernice – 32 bit
- dĺžka analógového/digitálneho signálu v bitoch – 12 bit
- maximálna frekvencia hodín – 48 MHz

- počet vstupov/výstupov – 71 I/O
- dostupné A/D kanály – 16
- počet časovačov – 6
- rozhrania – I2C, SPI, UART, CAN

Následne som kvôli rozširujúcej nadstavbe CANopen využil aj miktroprocesor LPC11C24 od spoločnosti NXP, ktorý má väčšiu podporu tohto rozšírenia.

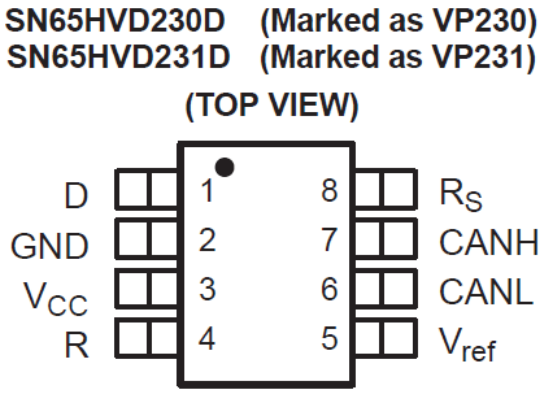
LPC11C24 je založený na ARM Cortex-M0, nízko nákladovej 32bitovej MCU skupine na tvorbu 8 alebo 16 bitových mikrokontrolérových aplikácii, poskytujúci výkon, nízku spotrebu, jednoduchú inštrukčnú sadu a adresovanie pamäte a obmedzenou veľkosťou kódu v porovnaní s danou 8/16 bitovou architektúrou. [15]

- 50MHz Cortex-M0 procesor
- 32KB/16KB Flash, 8KB SRAM
- CAN 2.0 B C_CAN kontróler z on-chip CANopen driver, integrovaný vysielač
- UART, 2 SPI & I^2C (FM+)
- 2x 16bit a 2x 32bit časovače s PWM/Match/Capture, 1x 24bit systémový časovač
- 12MHz interný RC oscilátor
- Power-On-Reset
- 8-kanálový veľmi precízny 10bitový ADC
- EME (Low Electromagnetic Emission) a EMI (high Emectromagnetic Immunity) CAN vysielač

5.4. CAN Budič

Je to vlastne prijímač a vysielač, ktorý prevádza data CAN na elektrické signály zbernice a naopak prijíma signály, ktoré mení na data.

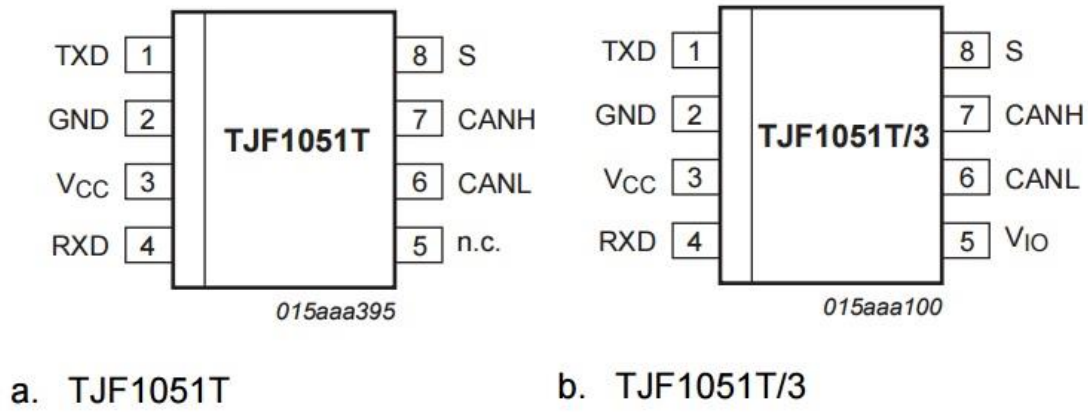
Vývojová doska FRDM – KE06Z obsahuje budič SN65HVD230D od spoločnosti Texas Instruments.



Obrázok 13: SN65HVD230D[6]

Vývojová doska LPC11C24 obsahuje CAN budič TJF1051 od firmy NXP Semiconductors.

- Low ElectroMagnetic Emission (EME)
- High ElectroMagnetic Imunity (EMI)



Obrázok 14: LPC budič TJF1051[14]

6. Nastavenie CANu a implementácia

6.1. Projekt a základné nastavenie KDS

Po spustení KDS vyberieme umiestnenie workspace. Následne vytvoríme nový projekt pomocou File => New => Kinetis Design Studio Project. Zadáme názov projektu, umiestnenie a presunieme sa do Devices, kde vyberieme typ procesoru (Processors => Kinetis E => KE06Z / KEAZ (48MHz) => MKE06Z128xxx4) tlačidlom Next. Cez ďalší Next povolíme Processor Expert a potvrdíme. V záložke Components, priečinku Components zmažeme všetky komponenty. Následne v záložke Project Explorer pravým kliknutím na vytvorený projekt zvolíme Build Project. V našom projekte by sa mal objaviť súbor Binaries. Debug nastavíme v Debug Configurations dvojklikom na GDB PEMicro Interface Debugging vytvoríme konfiguráciu. V záložke Debugger nastavíme Interface na OpenSDA Embedded Debug – USB Port, Port podľa pripojeného zariadenia a v Device Name vybrať typ procesoru (KE06Z128M4). Pomocou Apply potvrdíme zmeny. Týmto sú základné nastavenia pokryté.

6.2. Nastavenie CAN v Kinetis Design Studio

Nastavenie sa prevádza pomocou Processor Experta. V Components Library si vyberieme daný komponent, po prekliknutí na Component Inspector môžeme začať nastavovať. Po výbere komponentu CAN_LDD môžeme v nastaveniach zmeniť názov, povolíme prerušenia v podzložke Interrupt Service. V podzložke Settings nastavíme piny Rx (PTH2) a Tx (PTE7). Podzložka Timing nastavuje prenosovú rýchlosť. Pomocou CAN timing calculator v poli CAN bit timing zaklikneme Auto select v oboch časovacích segmentoch a v poli Bit rate nastavíme požadovanú rýchlosť.

V zložke Methods vyberieme metódy, ktoré chceme vygenerovať výberom z rolovacieho poľa pri danej metóde (generate code). V záložke Components – názov projektu => Generate Processor Expert Code vygenerujeme vybrané metódy a potvrdíme zmeny nastavenia. Generované kódy nájdeme v našom projekte v priečinku Generated_Code.

6.3. Nastavenie ďalších použitých komponentov

Pre komunikáciu s PC je potrebné nastaviť sériovú komunikáciu UART. Po pridaní komponentu z Components Library (Serial_LDD) si vyberieme Device (UART0-2), povolíme prerušenia. V položke Settings nastavíme Baud rate (9600), RxD (PTD6) a TxD(PTD7). V záložke Methods si vyberieme nami využívané metódy.

Najdôležitejším nastavením je nastavenie CPU. V Properties => Clock settings => System oscillator povolíme System oscillator, Clock source vyberieme External crystal a Clock frequency nastavíme na 8MHz. Oscillator operating mode – Low power. Clock source settings => ICS settings:

- ICS mode – FBE
- FLL output [MHz] – 40
- ICS output clock – 8

Clock configurations => System Clocks – Core, Bus a Timer Clock = 8.

Pre vytvorenie príkladu a otestovanie funkčnosti môžeme využiť komponenty typu RGB Led, prepínače atd.

6.4. Program pre odosielanie a prijímanie správ pomocou CAN

Program začína definíciou premenných, alokáciou miesta a textom odosielanej správy. Následne som nastavil ID, typ, odosielané dáta a veľkosť správy. Keďže na signalizáciu využívam RGB Led, musím túto ledku zhasnúť. Potom nasleduje samotné odosielanie správ. Metódou SendFrame odosielam dáta. To že sú odoslané indikuje premenná DataTxFlag typu bool, ktorá využíva udalosť (Event) OnFreeTxBuffer. Táto udalosť nastaví premennú DataTxFlag na TRUE, za predpokladu, že posledný bit správy bol odoslaný. Zelená signalizuje odosielanie správ. Po stlačení prepínača (SW1) rozsvietim modrú ledku a skočím do nekonečného cyklu while, kedy preruším odosielanie správ.

Pre prijímanie správ využívam metódu ReadFrame. Prijatie celej správy signalizuje premenná DataRxFlag typu bool, ktorá využíva Event OnFullRxBuffer. Metóda nastaví DataRxFlag na TRUE po prijatí kompletnej správy. Túto správu ukladám do dvojrozmerného poľa. Pole je potrebné na vypisovanie správ cez sériovú linku UART do PC. Zelená led signalizuje prijatie správ. Pri prijatí viac než 800 správ sa rozsvieti červená ledka. Po stlačení tlačidla 1 (SW1) rozsvietim modrú ledku a čakám na stlačenie tlačidla 2 (SW2), ktoré realizuje odosielanie správ z poľa do PC pomocou UART komunikácie metódou SendBlock.

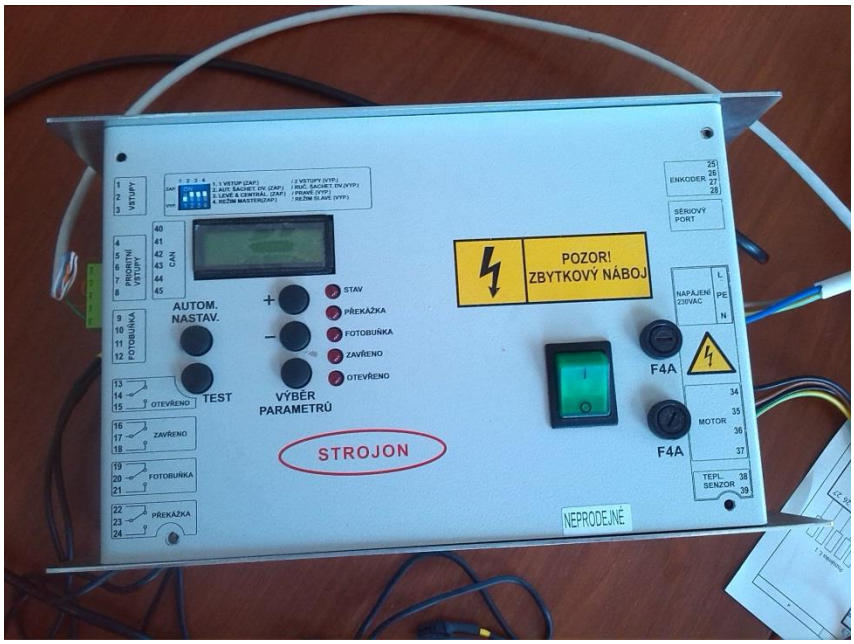
Použitím týchto dvoch aplikácií testujem na akú vzdialenosť dokážu zariadenia komunikovať maximálnou možnou rýchlosťou. V aplikácii na odosielanie nastavím počet správ na 200 a mením prenosovú rýchlosť podľa toho, či správy "zvládli" prenos, čím zistím dĺžku vedenia pre maximálnu možnú rýchlosť. Je potrebné nastaviť rovnakú prenosovú rýchlosť na všetkých zariadeniach.

6.5. Projekt a základné nastavenie LPCXpresso

Pre prácu v LPCXpresso je vhodné sa registrovať a aktivovať danú verziu programu. Vykonanie tohto kroku je popísané po v pracovnom okne po tom ako sa dostaneme do vývojového prostredia. Po spustení programu je potrebné vybrať priečinok na ukladanie projektov tzv. workspace. Pomocou Browse sa vyberá umiestnenie, prípadne vytvorenie priečinku na uloženie projektu. Potvrdením tlačidlom OK sa objaví vývojové prostredie, ktoré bližšie popísané v dokumente. V prvom rade je nutné načítať CMIS-CORE knižnicu pre náš mikrokontrolér. Pomocou Import project a vyhľadaním potrebnej knižnice pokračujeme k výberu konkrétneho projektu. Pomocou tlačidla Finish ukončíme import projektu. Nový projekt sa vytvorí pomocou wizarda, ktorý sa spustí po kliknutí na New project... Následne sa zobrazí výzva na výber nami používaného mikroprocesora a typu projektu (C, C++...). Potom zadáme názov projektu a cieľové umiestnenie. V ďalšom kroku upresníme typ mikrokontroléra. Pokračujeme pomocou Next na posledný list nastavení, kde dokončíme vytvorenie projektu tlačidlom Finish.

6.6. Komunikácia s priemyslovým CAN zariadením

V tejto časti je ukážka jednoduchej komunikácie s priemyslovým zariadením VTA-DOOR pomocou protokolu CANopen cez zbernicu CAN. VTA-DOOR je riadiaca jednotka výťahových dverí, ktorá je určená pre riadenie kabínového pohonu automatických dverí. Súčasťou dodávky kabínového pohonu je aj motor 230Vac. Nastavenie a identifikácia stavov je možná pomocou tlačidiel, prepínačov, LED indikátorov a LCD displeja priamo na riadiacej jednotke. Toto zariadenie je nastavené ako NMT slave.



Obrázok 15: Riadiaca jednotka výťahových dverí VTA

Obrázok 16: Kabínový pohon

6.6.1. Realizácia komunikácie s priemyslovým zariadením

Komunikáciu riešim pomocou mikrokontroléra LPC11C24, ktorý je v tomto systéme nadradený (NMT master). V aplikácii som vytvoril Object Dictionary podľa dodanej špecifikácie od firmy BetaControl. Následne som vytvoril funkcie, ktoré realizujú hlavné komunikačné objekty. Na jednoduchú aplikáciu využívam objekt TPDO1, ktorý má v OD namapovaný konfiguračný PDO, ktorý realizuje komunikáciu na prenos procesných dát medzi mikrokontrolérom a riadiacou jednotkou. Ďalšia nutná funkcia na to aby som pomocou objektov procesných dát mohol komunikovať s jednotkou je NMT správa, ktorou nastavím jednotku do operational módu (len v operational móde je možné odosielať PDO správy). Na čítanie a zápis záznamov v Object Dictionary používam funkcie SDO upload a SDO download.

Aplikácia vykonáva jednoduchú úlohu. Na začiatku nastavím riadiacu jednotku do operačného stavu pomocou NMT správy. Následne vyšlem SDO_upload správu, pomocou ktorej zistím v akom stave sú dvere. Ak sú zatvorené, vyšlem TPDO správu, ktorou otvorím dvere. Po otvorení dverí ich pomaly zatvorím, taktiež odoslaním TPDO správy s daným pokynom. V prípade, že sú dvere otvorené, TPDO správou vyšlem príkaz na ich zatvorenie.

Line	Arrival Time [ms]	ID [Hex]	Dir	Data | 0,	1,	2,	3,	4,	5,	6,	7]	Description
1	13:15:59.624	701	Rx	00								Boot-up Heartbeat
2	13:15:59.624	701	Rx	7F								Pre-Operational
3	13:16:09.527	701	Rx	7F								
5	11:27:30.512	000	Rx	01	01							NMT- Operational
6	11:27:30.513	601	Rx	40	01	20	01	00	00	00	00	SDO client-server
7	11:27:30.523	581	Rx	4F	01	20	01	02	00	00	00	SDO server-client
8	11:27:30.524	201	Rx	01	00	00	00	00	00	00	00	VTA RPDO1
9	11:27:30.532	181	Rx	02	0E							VTA TPDO1 - zatvorené
10	11:27:30.731	181	Rx	22	0E							VTA TPDO1 - otváram(zatvorené)
11	11:27:30.929	181	Rx	20	0E							
12	11:27:31.127	181	Rx	20	0E							VTA TPDO1 - otváram
33	11:27:35.285	181	Rx	21	0E							VTA TPDO1 - otváram(otvorené)
34	11:27:35.298	601	Rx	40	01	20	01	00	00	00	00	SDO client-server
35	11:27:35.305	581	Rx	4F	01	20	01	21	00	00	00	SDO server-client
36	11:27:35.306	201	Rx	02	00	00	00	00	00	00	00	VTA RPDO1
37	11:27:35.484	181	Rx	41	0E							VTA TPDO1 - zatváram(otvorené)
38	11:27:35.682	181	Rx	40	0E							
39	11:27:35.880	181	Rx	40	0E							VTA TPDO1 - zatváram
60	11:27:40.037	181	Rx	42	0E							VTA TPDO1 - zatváram(zatvorené)
61	11:27:40.215	701	Rx	05								Operational Heartbeat
62	11:27:40.235	181	Rx	42	0E							VTA TPDO1 - zatváram(zatvorené)
104	11:28:00.080	701	Rx	7F								Pre-Operational Heartbeat

Obrázok 17: Záznam komunikácie LPC-VTA

6.7. Testovacia aplikácia

Testovaciu aplikáciu som vytváral v konzolovom prostredí Microsoft Visual Studio 2008. Táto aplikácia odosiela cez sériové porty informácie o prenosovej rýchlosti zbernice CAN a požadovaný počet CAN správ.

```
********************************************************************************************************
********************************************* CAN SPEED TESTER *****************************************
********************************************************************************************************
Opening serial OUT port...OK
Opening serial IN port...OK

Serial Out Communication state
BaudRate = 9600, ByteSize = 8, Parity = 0, StopBits = 1

Serial In Communication state
BaudRate = 9600, ByteSize = 8, Parity = 0, StopBits = 1



Sedning Controller?             SET: 'L' for LPC11C24 or 'F' for FRDM-KE06Z
SET: l

You set Controller: LPC11C24

OK
Set CAN bit rate:               (possible: 1000, 800, 500, 250, 125)kbit per second
bit rate: 500

Set number of messages
Count: 15

buff is now: 15
2
F
1
Reading bytes...
1 bytes readen

e


F
1
5
Closing serial OUT port...OK
Closing serial In port...OK
Press any key to continue . . .
```

Obrázok 18: Konzolový výpis

Aplikácia je navrhnutá na komunikáciu s LPC mikrokontrolérmi, kde je možnosť pomocou funkcie baudratecalculate prepočítať prenosovú rýchlosť. U FRDM-KE06Z je nutné túto funkciu vytvoriť cez registre. Táto možnosť sa mi nezdala spoľahlivá, tak som sa sústredil len na komunikáciu s LPC.

Po odoslaní správ pomocou dvoch sériových liniek, LPC mikrokontroléry spracujú prijaté dáta a prepočítajú prenosovú rýchlosť. Mikrokontrolér, ktorý bude odosielať dáta nastaví počet správ. Po odoslaní všetkých správ, prijímajúci mikrokontrolér odošle späť do konzoly (pomocou UART) počet prijatých CAN správ. Porovnaním odoslaných a prijatých dát, môžeme vyhodnotiť komunikáciu.

6.8. Meranie rýchlosti komunikácie v závislosti na vzdialenosti a komunikačnej rýchlosti

Meranie som realizoval na 24 žilovom kábli o dĺžke 30 metrov, ktorý som podľa potrieb prepájal. K dispozícii som mal dva mikrokontroléry LPC a dva FRDM-KE06Z. Odosielal som 200 CAN správ. V nasledujúcich tabuľkách silne zelená farba zobrazuje bezproblémový prenos, čiže počet odoslaných a prijatých správ je rovnaký. Svetlejšia zelená znázorňuje pár chýb v prenose. Svetlá farba označuje, že množstvo prenesených dát bolo malé a biela farba, že sa neodoslala žiadna správa.

Tabuľka 2: Komunikácia KE06Z - KE06Z

	Vzdialenosť [m]									
	30	60	90	120	150	180	210	240	270	300
1Mb/s										
800kb/s										
500kb/s										
250kb/s										
125kb/s										

Tabuľka 3: Komunikácia LPC - KE06Z

	Vzdialenosť [m]									
	30	60	90	120	150	180	210	240	270	300
1Mb/s										
800kb/s										
500kb/s										
250kb/s										
125kb/s										

Tabuľka 4: Komunikácia LPC - LPC

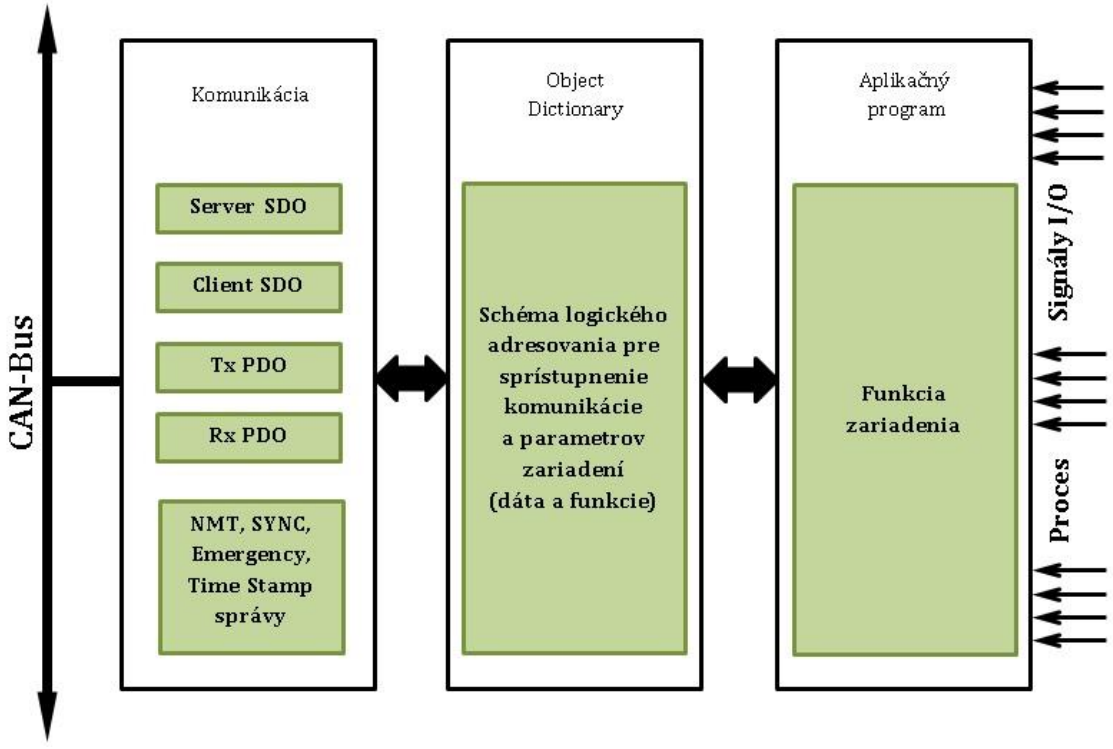
	Vzdialenosť [m]									
	30	60	90	120	150	180	210	240	270	300
1Mb/s										
800kb/s										
500kb/s										
250kb/s										
125kb/s										

Z nameraných hodnôt vyplýva, že komunikácia medzi kombináciou LPC a KE06Z nie je najspoľahlivejšia. Dôvodom je pravdepodobne rozličné zakončenie zbernice. Najväčšiu vzdialenosť na rýchlosti 1Mb/s sa mi podarilo namerať s mikrokontrolérmi KE06Z, o dĺžke zbernice 38 metrov. Na rýchlosti 500kb/s približne 93 metrov.

9. CANopen

CANopen je druh sieťového protokolu založeného na zbernici CAN a využíva sa vo vozidlách, automatizácii budov, priemyselných strojoch, zdravotníckych zariadeniach, námorníckych aplikáciách a iné. Okrem broadcastingu taktiež umožňuje peer to peer prenos dát medzi všetkými CANopen uzlami. Funkcia sieťového manažmentu špecifikovaná v CANopen zjednodušuje návrh projektu. Užívatelia môžu taktiež implementovať a diagnostikovať CANopen sieť štandardnými mechanizmami pre siete start-up a error management (správu chýb). Akékoľvek CANopen zariadenie môže efektívne pristupovať alebo získať podmienky týkajúce sa I/O (Input/Output) hodnôt a stav uzlov ostatných zariadení siete. Zariadenie CANopen môžeme obecne formovať do troch častí[13]:

- Komunikácia
- Object Dictionary
- Aplikačný program



Obrázok 19: CANopen zariadenie

7.1. Komunikácia

Táto časť poskytuje niekoľko komunikačných objektov a vhodné funkcie na prenos CANopen správ cez CAN. Objektmi môžu byť NMT (Network Management Objects), SDO (Service Data Object), PDO (Process Data Object), SYNC (Synchronous Objects) a iné. Každý komunikačný objekt má svoj komunikačný model a funkciu. Nezáleží, ktorý komunikačný objekt je využívaný, odosielaná správa sa musí riadiť dátovým rámcom definovaným v špecifikácii CAN 2.0A.[13]

Tabuľka 5: Obsah štandardného CANopen uzla

	COB-ID	RTR	Dĺžka dát	Dáta
Dĺžka	11 bitov	1 bit	4 bity	0-8 bytov

RTR = 1 – táto správa sa využíva na žiadosť o vzdialené vysielanie (remote-transmit request). V tomto prípade 8 dátových bytov je nevyužitých.
Dĺžka dát indikuje počet platných dát uložených v dátovom poli.
Dáta – obsahuje dáta správy.
COB-ID (Communication object ID) sa skladá zo 4 bitov funkčného kódu a 7 bitov ID uzlu.

Tabuľka 6: COB-ID

	Funkčný kód				ID uzlu						
bit	10	9	8	7	6	5	4	3	2	1	0

COB-ID sú definované na rozoznávanie, odkiaľ správy prišli alebo kam sa musia odoslať. Taktiež sa používajú na rozlíšenie funkcie prijatých alebo odosielaných správ a rozhoduje o priorite odosielaných správ každého uzla v sieti. CAN správa s nižšou hodnotou COB-ID má vyššiu prioritu. V CANopen špecifikácii sú niektoré COB-ID rezervované pre špeciále komunikačné objekty a nemôžu byť definované užívateľmi svojvoľne.[13]

Tabuľka 7: Rezervované COB-ID[13]

Rezervované COB-ID [Hex]	Pouzívané objektmi
0	NMT
1	Rezervované
80	SYNC
81-FF	EMERGENCY
100	TIME STAMP
101-180	Rezervované
581-5FF	Transmit-SDO
601-67F	Receive-SDO
6E0	Rezervované
701-77F	NMT Error Control
780-7FF	Rezervované

Tabuľka 8: COB-ID pre užívateľov[13]

Funkčný kód (Bit10-Bit7)	(Bit6-Bit0)	Komunikačný Objekt
0000	0000000	NMT
0001	0000000	SYNC
0010	0000000	TIME STAMP
0001	ID uzlu	EMERGENCY
0011/0101/0111/1001	ID uzlu	TxPDO1/2/3/4
0100/0110/1000/1010	ID uzlu	RxPDO1/2/3/4
1011	ID uzlu	TxSDO
1100	ID uzlu	RxSDO
1110	ID uzlu	NMT Error Control

7.2. Object dictionary (OD)

Object Dictionary je druh tabuľky, ktorá obsahuje všetky sieti prístupné informácie. Každý CANopen uzol musí obsahovať vlastný OD, ktorý pozostává zo záznamov obsahujúce popis nastavení CANopen a funkcie uzlu, v ktorom je uložený. Každý záznam má Index využívaný na prístup do záznamu. Index je 16 bitové číslo, ktoré udáva počet záznamov (max. 65 534). Každý záznam môže obsahovať až 256 pod záznamov (Subentry), na ktoré odkazuje 8 bitový Subindex. Každý záznam má minimálne jeden pod záznam.

Nie všetky záznamy v Object Dictionary sú implementované alebo použité. Bežne sa v praxi na popis Indexov a Subindexov v OD používa hexadecimálny zápis. Záznamy, ktoré ukladajú len jednu hodnotu, obsahujú len jeden pod záznam na Subindexe 00h. Záznamy ukladajúce viac než jednu hodnotu musia mať pre každú hodnotu pod záznam a ukladať číslo najvyššieho pod záznamu na Subindex 00h. Záznamy môžu byť čítané alebo prepisované ostatnými CANopen uzlami.[12]

Tabuľka 9: Príklad záznamov v OD

Index	SubIndex	Typ	Význam
1000h	0	UINT32	Device Type Information
1001h	0	UINT8	Error Register
1200h	0	UINT8	Number of entries (2)
	1	UINT32	COB-ID Client-Server
	2	UINT32	COB-ID Server-Client

Object Dictionary pozostáva z rôznych dátových typov, ktorých popisy sú taktiež uložené v OD. Všetky možné Indexy (65 536) sú rozdelené na sekcie.

Tabuľka 10: Rozdelenie Indexov

Rozsah Indexov	Popis
0000h	Rezervované
0001h-0FFFh	Dátové typy
1000h-1FFFh	Komunikačné záznamy
2000h-5FFFh	Záznamy konkrétneho výrobca
6000h-9FFFh	Parametre profilu zariadenia
A000h-FFFFh	Rezervované

7.2.1. Dátové typy

Object Dictionary môže ukladať štandardné / preddefinované a továrne definované dátové typy. Taktiež špecifikácia CANopen definuje dve základné triedy dátových typov, Standard a Complex. Usporiadanie a rozsahy Indexov v OD sú znázornené v dokumente. Pri realizácii CANopen uzla je možné si definovať vlastný komplexný dátový typ v časti Manufacturer Complex Data Types. Komplexné dátové typy pozostávajú z jedného alebo viacerých štandardných dátových typov zoskupených dohromady. Je to obdoba štruktúr v programovacom jazyku C. [12]

7.2.2. Komunikačné záznamy

Tieto záznamy v Object Dictionary popisujú väčšinu komunikačných CANopen aspektov používaných uzlom. Veľa záznamov môže byť prepisovaná, čím umožňuje konfigurovať uzol, ostatnými zariadeniami na sieti. Komunikačné záznamy obsadzujú Indexy OD v rozsahu 1000h-1FFFh. Niektoré záznamy sú povinné, a Object Dictionary ich musí obsahovať. Povinné (Mandatory) záznamy sú[12]:

- Device Type (1000h) [32b] – krátky popis vlastností uzla. Napríklad či sa jedná o digitálny vstupný/výstupný modul sú vstupy alebo výstupy implementované.
- Error Register (1001h) [8b] - indikuje rôzne druhy vzniknutých chýb v zariadení.
- Guard Time (100Ch) [16b] - konkretizuje, ako často je odosielaná žiadosť nadradeným uzlom, alebo musí byť prijatý uzlom.
- Life Time Factor (100Dh) [8b] - tento záznam sa musí implementovať, ak sa nepoužíva heartbeat. LTF spolupracuje s Guard Time, kde špecifikuje koľko násobkov Guard Time musí prejsť bez prenosu od mastra alebo prijatí odpovedí od podriadeného uzla predtým, než sa vygeneruje chybový podmienka.
- Producer Heartbeat Time (1017h) [32b] – ak sa nepoužíva node guarding tak musí byť implementovaný heartbeat. Tento záznam udáva, ako často môže uzol odosielať heartbeat správu. Pri nastavení na nulu sa zablokuje odosielanie heartbeat.
- Identity Object (1018h) - poskytuje identifikačné informácie o uzle. Minimálne musí obsahovať pridelený ID predajcu, ktoré je unikátne u každého obchodníka.

7.2.3. Záznamy konkrétneho výrobca

V Object Dictionary používajú Indexy v rozsahu 2000h-5FFFh a sú úplne otvorené pre špecifické aplikácie. Sem sa ukladajú dáta alebo konfigurácie aplikácií, ktoré sú mimo CANopen štandard.

7.2.4. Parametre profilu zariadenia

Špecifikácia CANopen ponúka celú radu komunikačných služieb. Ako náhle je špecifický uzol implementovaný, návrhár uzla (alebo sieť, kde sa bude používať) má určiť, ktoré z týchto služieb sa používajú a ako. Device Profile špecifikuje premenné procesných dát, uzol pozná štandardné konfiguračné a komunikačné nastavenia. [12]

7.3. Aplikačný program

Aplikačná časť riadi všetky funkcie zariadenia. Jedná sa o most medzi Object Dictionary a praktickým procesom.

7.4. Service Data Objects (SDO)

Tieto objekty poskytujú prístup k záznamom Object Dictionary. Pomocou komunikačnej metódy SDO sa stanovuje komunikačný most medzi dvoma zariadeniami (peer to peer). SDO prenos využíva vzťah klient - server.

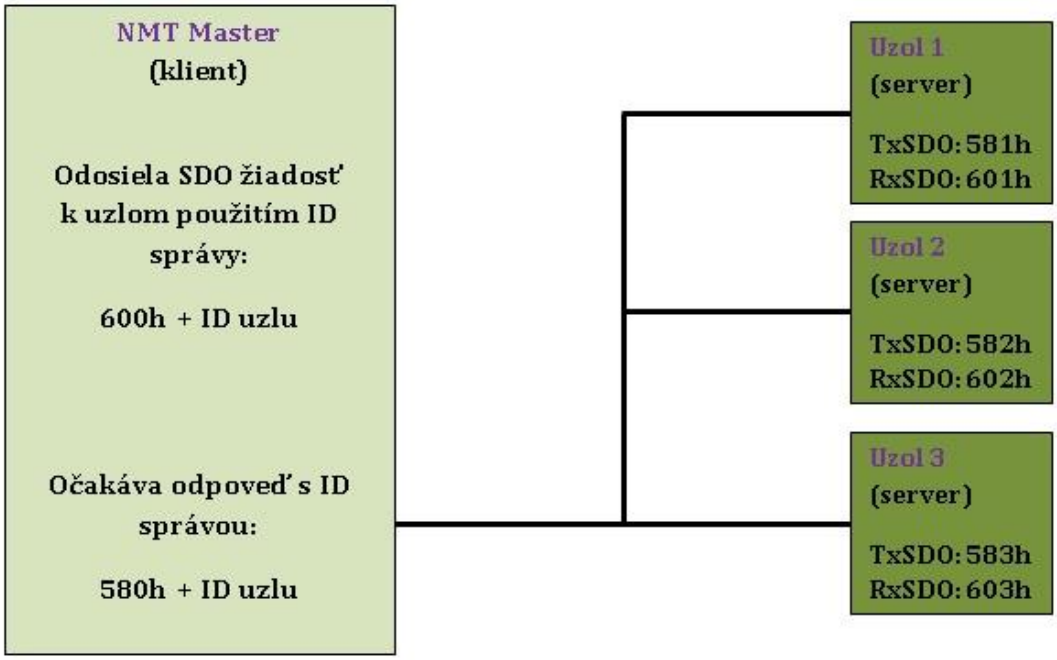
Každý CANopen uzol má okrem Object Dictionary implementovaný server, ktorý spracováva požiadavky na zápis a čítanie do a z OD. Konfiguračný nástroj alebo master sa chová ako klient k serveru a môže odosielať žiadosti na zápis alebo čítanie. Príklad, master odošle žiadosť: uzol číslo 2, potrebujem informácie ktoré máš na Indexe 100Ah, Subindexe 00h. Uzol 2 rozozná žiadosť a odpovedá: Ktokoľvek sa pýtal, tu sú žiadané dáta.

Service Data Objects používajú na komunikáciu medzi zariadeniami dva druhy identifikátorov (COB-ID). Identifikátor správy, ktorý sa využíva na odosielanie žiadosti určitému uzlu sa určuje pričítaním ID uzlu (1-127) k základnej adrese 600h. Čiže adresy 601h-67Fh sa poskytujú 127 kanálov od klienta k 127 serverom.

Identifikátor správy využívaný na odoslanie odpovede z každého uzla späť ku klientovi, ktorý odoslal žiadosť sa určuje pričítaním ID uzlu (1-127) k základnej adrese 580h. To znamená, že adresy v rozsahu 581h-5FFh poskytujú 127 kanálov na odpovede serverov späť ku klientovi.

Celá SDO komunikácia bola založená na myšlienke, že len jeden uzol v systéme potrebuje moc na prístup k úplne všetkým záznamom OD v každom jednom uzle.

Všetky SDO správy (žiadosti / odpovede) majú 8 bytov dát, kde prvý byte je



Obrázok 20: SDO komunikačné správy

takzvaný špecifikátor (specifier). Najvyššie bity informujú o type správy, čiže čítanie, zápis alebo zlyhanie (abort - indikuje chybu). Ostatné bity indikujú akýsi formát správy, to znamená či sa dáta prenášajú v jednej správe (expedited transfer), alebo sú rozdelené do viacerých správ (segmented transfer) prípadne prenos veľkých dátových blokov (block transfer).

Byty 2 až 4 predstavujú multiplexor – kombinácia 16 bitového Indexu a 8bitového Subindexu, identifikujúci záznam OD, ktorý je prístupný s týmto SDO. Druhý byte obsahuje nižší byte 16bitového Indexu, tretí byte vyšší byte a štvrtý byte 8bitový Subindex.

Zvyšné byty (5-8) sa využívajú na prenos dát. V prípade, že dáta na prenos majú 4 alebo menej bytov, jedná sa o expedited správu, pri väčšom počte bytov segmented.

Pri segmentovom prenose správ, prvá správa neobsahuje žiadne dáta, ale informujú koľko bude celkom prenášaných dát. Potom každý prenášaný segment v prvom dátovom byte informuje, či je posledný, a ak je, tak koľko dátových bajtov prenáša v aktuálnej správe, ktoré patria k prevodu.

Kedykoľvek počas prenosu môže použitím SDO abort správy, niekto z komunikačnej dvojice zrušiť prenos. [12][13]

7.4.1. SDO Upload vs. Download

Tieto výrazy v CANopen sú trochu mätúce. SDO upload predstavuje prístup pre čítanie a SDO download prístup pre zápis. Preto je výhodnejšie používať výrazy "SDO Read Access" (čítanie) a "SDO Write Access" (zápis).[12]

7.4.1.1. SDO žiadosť o zápis „Download – Request"

Klient odosiela žiadosť na SDO server použitím CAN identifikátora 600h plus ID uzla, do ktorého chce zapisovať. Typicky sa jedná o pokus nakonfigurovať CANopen podradený (slave) uzol.

	1. Byte								2. Byte	3. Byte	4. Byte	5.-8. Byte
bit	7	6	5	4	3	2	1	0	Index L	Index H	Subindex	Prenášané dáta alebo rezervované
	0	0	1	x = 0	n		e	s				
	ccs											

Obrázok 21: SDO Download-Request

Klient → Server

ccs: Client Command specifier
e: pre prenos typu expedited = 1
s: ak je zadaná veľkosť dát = 1
n: ak e=s=1, počet dátových bitov, ktoré neobsahujú dáta
x: rezervované

7.4.1.2. SDO odpoveď na žiadosť o zápis „Download - Response"

Túto správu posiela späť ako odpoveď SDO server klientovi, keď potvrdí predošlú žiadosť. Použitím CAN identifikátora v tvare 580h plus ID uzlu SDO servera.

	1. Byte								2. Byte	3. Byte	4. Byte	5.-8. Byte
bit	7	6	5	4	3	2	1	0	Index L	Index H	Subindex	Rezervované
	0	1	1	x = 0								
	scs											

Obrázok 22: SDO Download-Response

Klient ← Server

scs: Server Command Specifier
x: rezervované

7.4.1.3. SDO žiadosť o čítanie „Upload - Request“

Klient odosiela žiadosť na SDO server použitím CAN identifikátora 600h plus ID uzla, z ktorého chce čítať. Typicky sa jedná o pokus nakonfigurovať CANopen podradený (slave) uzol.

	1. Byte								2. Byte	3. Byte	4. Byte	5.-8. Byte
bit	7	6	5	4	3	2	1	0	Index L	Index H	Subindex	Rezervované
	0	1	0	x = 0								
	ccs											

Obrázok 23: SDO Upload-Request

Klient → Server

ccs: Client Command specifier
x: rezervované

7.4.1.4. SDO odpoveď na žiadosť o čítanie „Upload - Response“

Túto správu posiela späť ako odpoveď SDO server klientovi, keď potvrdí predošlú žiadosť. Pri použití expedited prenosu prečítané dáta z OD sú súčasťou správy, inak sa použije segmentový prenos. Použitím CAN identifikátora v tvare 580h plus ID uzlu SDO servera.

	1. Byte								2. Byte	3. Byte	4. Byte	5.-8. Byte
bit	7	6	5	4	3	2	1	0	Index L	Index H	Subindex	Prenášané dáta alebo rezervované
	0	1	0	x = 0	n		e	s				
	scs											

Obrázok 24: SDO Upload-Response

Klient ← Server

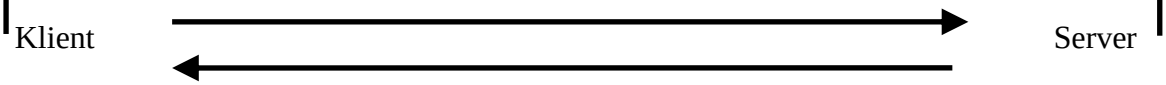
scs: Server Command Specifier
e: pre prenos typu expedited = 1
s: ak je zadaná veľkosť dát = 1
n: ak e=s=1, počet dátových bitov, ktoré neobsahujú dáta
x: rezervované

7.4.1.5. SDO Abort Transfer (zlyhanie prenosu)

Kedykoľvek môže server alebo klient zrušiť prenos použitím SDO abort správy. Error code (kód chyby) informuje prečo nastalo zrušenie prenosu. Typické chyby sú napríklad nesprávna dĺžka dát alebo záznam OD nie je implementovaný a iné. Zoznam možných chýb je uvedený v dokumente. [12]

	1. Byte								2. Byte	3. Byte	4. Byte	5.-8. Byte
bit	7	6	5	4	3	2	1	0	Index L	Index H	Subindex	Error Code
	1	0	0	x = 0								
	cs											

Obrázok 25: Abort SDO



cs: Command Specifier
x: rezervované

7.5. Process Data Object (PDO)

Objekty procesných dát sú prispôsobené na efektívny prenos cez zbernicu CAN. Môžu byť odosielané na časovač, alebo odpoveď na SYSN správu alebo odpoveď na udalosť(podľa zmien napr. digitálneho vstupu).

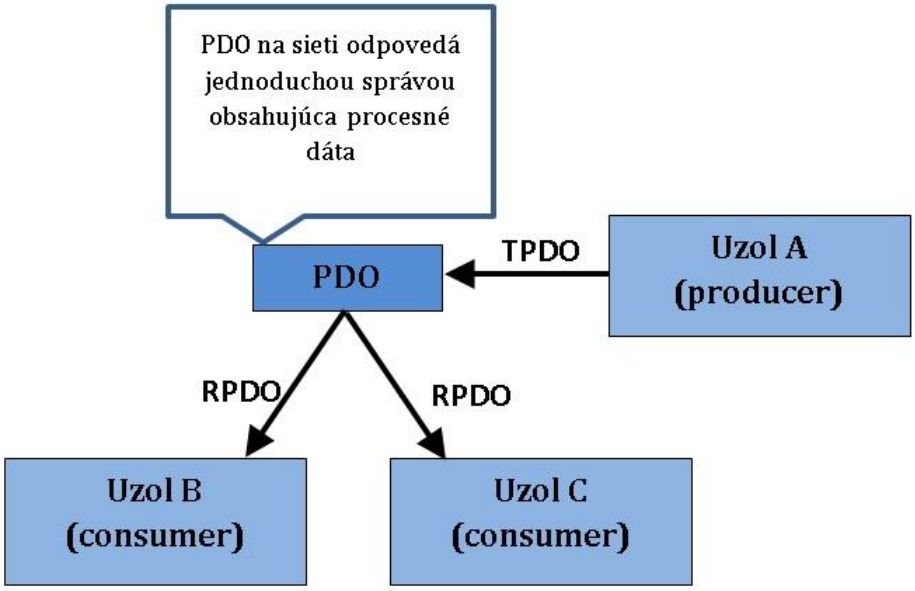
Na rozdiel od SDO, kde je možné odoslať maximálne 4Byty v jednej správe, PDO môže využiť celých 8 dátových bytov naraz a taktiež má vyššiu prioritu.

PDO príjem alebo prenos dát je riešený pomocou komunikačného modelu typu producer/consumer (taktiež nazývaný push/pull model). To znamená, že odosielaná správa uzlom (producer) môže byť prijatá ostatnými uzlami (consumers). Tento princíp je veľmi efektívny na prenos real-time dát. Správy sa delia na TPDO správy, ktoré sú na odosielanie zo zariadenia, a RPDO, ktoré sú prijaté a zapisujú do Object Dictionary zariadenia. Tieto správy sa od seba mierne líšia, TPDO správy vyžaduje viac parametrov než RPDO správy. [12][13]

	COB-ID										RTR	Veľkosť dát	8 Dátových bytov							
	Funkčný kód				ID uzla															
bit	10	9	8	7	6	5	4	3	2	1	0	0	8	0	1	2	3	4	5	6	7

Obrázok 26: Formát PDO správy

Možnosť komunikovať pomocou PDO správ je len v Operational stave.



Obrázok 27: Príklad PDO správ

7.5.1. PDO Linking

Z pohľadu siete nie je PDO nič iné ako správa s identifikátorom a 8 dátových bytov. Skoro všetko je v CANopen konfigurovateľné, vrátanie identifikátorov PDO správ. CANopen má pre PDO vytvorený takzvaný "pre-defined" komunikačný, ktorý stanovuje, aké COB-ID by malo byť použité pre ktorý uzol. [12]

Tabuľka 11: Pre-defined PDO

PDO	CAN ID	
	od	do
TPDO1	181h	1FFh
RPDO1	201h	27Fh
TPDO2	281h	2FFh
RPDO2	301h	37Fh
TPDO3	381h	3FFh
RPDO3	401h	47Fh
TPDO4	481h	4FFh
RPDO4	501h	57Fh

CANopen podporuje štyri hlavné spúšťacie metódy prenosu:

- Event driven (Change-Of-State (COS)) – odosiela TPDO správy ak sa zmenia procesné dáta.
- Time driven – TPDO sa odosiela v stanovenom čase. Tento čas sa nastavuje v Event Timer
- Individual polling – táto metóda je implementovaná ako opytovací mechanizmus (správa sa používa na vyvolanie uzla na uskutočnenie odoslania TPDO).

- Synchronized, group polling – využíva synchonizačný signál SYNC. SYNC signál je špecifická správa bez dát, používaná len za cieľom synchronizácie. Po odoslaní synchonizačnej správy, zariadenia vyšlú TPDO správu.

7.5.2. PDO mapping

PDO mapovanie obstaráva rozhranie medzi PDO správami a reálnymi I/O dátami v CANopen zariadení. Definuje význam každého bytu v PDO správe, ktoré môžu byť menené pomocou SDO správy. Všetky namapované PDO objekty sú umiestnené v Communication Profile Area. [13]

TPDO správy sa mapujú od indexu 1A00h a RPDO o indexu 1600h.

			Namapovaný TPDO		
Index	Subindex	Typ	Index	Subindex	Veľkosť
1A00h					
	0	UINT8	2000	01	08
	1	UINT8	2000	02	08

TPDO1	Index	Subindex	Typ	Hodnota
	2000	0	UINT8	=2
		1	UINT8	8bit hodnota
		2	UINT8	8bit hodnota

Obrázok 28: Príklad mapovania TPDO1 v OD

Všetky PDO správy sa musia týmto spôsobom namapovať. Z obrázku je zrejmé, že mapovanie pozostáva z Indexu, Subindexu a veľkosti dát daného PDO objektu. [17]

7.5.3. Network Management (NMT)

NMT protokoly sa využívajú na vydávanie príkazov stavu zariadenia (napríklad na zapnutie a vypnutie zariadenia), detekciu vzdialeného „bootup“ zariadenia a chybové podmienky. Zmena stavu zariadenia sa vykonáva pomocou NMT mastra. CAN rámec obsahuje COB-ID a dva dátové byty. COB-ID je rovný nule. Prvý dátový byte určuje požadovaný stav. Druhý dátový byte obsahuje ID uzla, ktorého sa to týka. Ak sa do druhého bytu zadá nula, požadovaný stav sa odošle každému na zbernici.[12] [16]

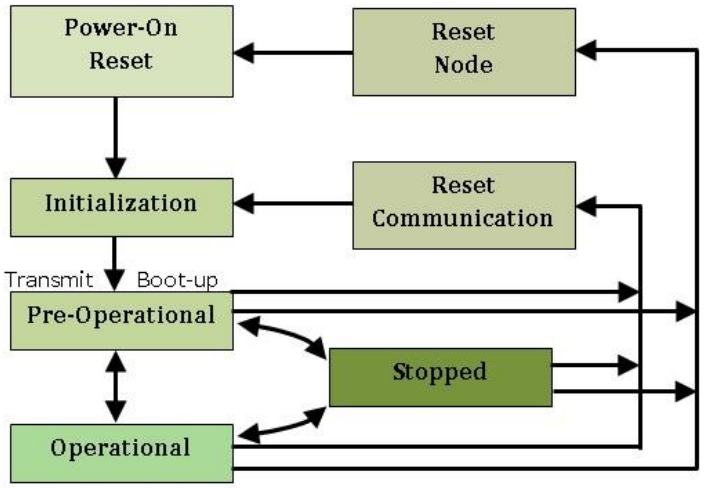
COB-ID	1. Dátový byte	2. Dátový byte
0x000	Požadovaný stav	ID uzla

Obrázok 29: Rámec NMT správy

Tabuľka 12: NMT stavy

NMT stavový kód	Mód
0x01	operational
0x02	stopped
0x80	pre-operational
0x81	reset node
0x82	reset communication

Každý NMT slave musí mať implementovaný NMT state machine, ktorý umožňuje zmeniť prevádzkový stav. Do niektorých stavov sa môžu podradené uzly nastavovať sami, ostatné stavy sa nastavujú po prijatí správy od NMT mastra.



Obrázok 30: Stavy NMT Slave

Po zapnutí napájania CANopen podradený uzol vychádza z "Power-On Reset" a postupuje do inicializačnej časti. Inicializuje celú aplikáciu a rozhrania CAN/CANopen a komunikáciu. Po skončení inicializácie sa uzol pokúsi odoslať boot-up správu. Po úspešnom odoslaní sa uzol prepne do Pre-Operational stavu. Tento stav je východiskový. Z tohto stavu sa pomocou správ NMT mastra prechádza do ostatných.

Reset Communication stav realizuje reštart CAN/CANopen komunikačného rozhrania.

Reset Node reštartuje celý uzol s periférami a softvérom.

Obe resetovacie stavy vyšlú novú boot-up správu o výsledku a nastaví sa späť do Pre-Operational stavu.[12][13]

	Initializing	Pre-Operational	Operational	Stopped
Boot-Up				
SDO				
Emergency				
SYNC/TIME				
Heartbeat/ Nodeguard				
PDO				
NMT				

Obrázok 31: Pracovné stavy

7.5.4. Heartbeat protocol

Tento protokol sa využíva na sledovanie zariadení v sieti a kontrolovať či sú aktívne. Heartbeat producer (väčšinou podradené zariadenie) odosiela periodicky správy pozostávajúce z COB-ID a jedného dátového bytu. Dátový byte informuje o stave zariadenia. Funkčný kód COB-ID má hodnotu 0x700h. Tento protokol oproti protokolu Node Guarding zaberá menšiu šírku pásma, viac flexibilnejší a bezpečnejší. [16]

COB-ID	1. Dáta byte
0x700h + ID uzlu	stav

Obrázok 32: Heartbeat rámec

Tabuľka 13: Stavy heartbeat

Kód NMT stavu	Význam stavu
0x00	Boot up
0x04	Stopped
0x05	Operational
0x7F	Pre-Operational

9. Záver

Cieľom tejto bakalárskej práce bolo oboznámiť sa s protokolom CAN a jeho rozširujúcou nadstavbou CANopen. Pomocou týchto vedomostí vytvoriť aplikácie, ktoré realizujú komunikáciu medzi CAN zariadeniami a porovnať zistené informácie s teoretickými predpokladmi.

Vytvoril som dielčie aplikácie na prenos a príjem správ po zbernici CAN pre dva rôzne mikrokontroléry. Taktiež aplikáciu na testovanie rýchlosti komunikácie v závislosti na vzdialenosti a komunikačnej rýchlosti. Po oboznámení sa s CANopen som vytvoril jednoduchú aplikáciu na prenos typických správ tejto nadstavby, ktorá realizovala komunikáciu s priemyslovým zariadením VTA-DOOR. Meraním som otestoval rýchlosti v závislosti na dĺžke zbernice. Z tohto merania vyplýva, že mikrokontroléry LPC11C24 dokážu komunikovať na väčšie vzdialenosti než vývojové kity FRDM-KE06Z alebo ich kombinácia, pri ktorej sa vyskytovali najčastejšie chyby. V tejto práci som využíval aj sériovú komunikáciu na prenos dát medzi mikrokontrolérmi a počítačom a USB-CAN adaptér na diagnostiku siete.

9. Literatúra

[1] ING. ZAVIDČÁK, Miroslav. *CAN - popis stroktury*. [online].2004 [cit. 2015-01-05]. Dostupné z: http://www.hw.cz/navrh-obvodu/rozhrani/can-popis-struktury.html

[2] KOCOUREK, P. *Sběrnicové systémy letadel: Controller Area Network (CAN).* [online]. 2008 [cit. 2015-01-05]. Dostupné z: http://measure.feld.cvut.cz/cs/system/files/files/cs/vyuka/predmety/x38ssl/CANPopis.pdf

[3] ING. TARABA, Radek. *Aplikování sběrnice CAN*. [online]. 2004 [cit. 2015-01-05]. Dostupné z:http://www.hw.cz/navrh-obvodu/rozhrani/aplikovani-sbernice-can.html

[4] ING. POLÁK, Karel. *Sběrnice CAN*. [online]. 2003 [cit. 2015-01-05]. Dostupné z: http://www.elektrorevue.cz/clanky/03021/index.html

[5] LEPKOWSKI, J a B WOLFE. ON SEMICONDUCTOR. *EMI/ESD protection solutions for the CAN bus.* 2005. CAN in Automation

[6] TEXAS INSTRUMENTS. *3.3-V CAN Transceivers*. Texas, 2011, 33 s.

[7] MOUSER ELECTRONICS. *Freescale Semiconductor MKE06Z128VLK4* [online]. [cit. 2015-01-05]. Dostupné z: http://cz.mouser.com/ProductDetail/Freescale-Semiconductor/MKE06Z128VLK4/?qs=CteNkmuaR6ykEuCdeDKG9w%3D%3D

[8] FREESCALE SEMICONDUCTOR. *FRDM-KE06Z: Freescale Freedom Development Platform for Kinetis KE06 MCUs* [online]. [cit. 2015-01-05]. Dostupné z: http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=FRDM-KE06Z

[9] ST. *NUCLEO-F103RB: STM32 Nucleo development board for STM32 F1 series* [online]. [cit. 2015-01-05]. Dostupné z: http://www.st.com/web/catalog/tools/FM116/SC959/SS1532/LN1847/PF259875

[10] BERAN, Martin. ÚSTAV AUTOMOBILNÍHO A DOPRAVNÍHO INŽENÝRSTVÍ. *Dátové sběrnice CAN* [online]. Brno, Česká republika [cit. 2015-01-05]. Dostupné z: http://www.iae.fme.vutbr.cz/userfiles/beran/files/Datov%C3%A1%20sb%C4%9Brnice%20CAN.pdf

[11] ELEMENT 14 COMMUNITY. *Kinetis Design Studio Integrated Development Environment* [online]. [cit. 2015-01-05]. Dostupné z: http://www.element14.com/community/docs/DOC-67925/l/kinetis-design-studio-integrated-development-environment#tools

[12] PFEIFFER,, Andrew AYRE a Christian KEYDEL. *Embedded Networking with CAN and CANopen* [online]. [cit. 2015-05-25]. Dostupné z: http://emanager.srmuniv.ac.in/elibrary/temp/CAN_and_CANopen.pdf

[13] IPC DAS. *I-7232D CANopen/Modbus RTU Gateway* [online]. [cit. 2015-05-25]. Dostupné z: http://www.bb-elec.com/Products/Manuals/i-7232dusermanual.pdf

[14] NXP SEMICONDUCTORS. *TJF1051: High-speed CAN transceiver* [online]. [cit. 2015-05-25]. Dostupné z: http://www.nxp.com/documents/data_sheet/TJF1051.pdf

[15] NXP SEMICONDUCTORS. *NXP Announces Industry's First Integrated CAN `` Transceiver Microcontroller Solution* [online]. [cit. 2015-05-25]. Dostupné z: http://www.nxp.com/news/press-releases/2011/01/nxp-announces-industry-s-first-integrated-can-transceiver-microcontroller-solution.html

[16] CANopen. 2001-. *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation [cit. 2015-05-25]. Dostupné z: http://en.wikipedia.org/wiki/CANopen

[17] PFIEFFER, Olaf. *Embedded Networking With CANopen* [online]. [cit. 2015-05-25]. Dostupné z: http://www.esacademy.com/en/library/technical-articles-and-documents/can-and-canopen/embedded-networking-with-canopen.html

Prílohy:
Príloha1: DVD